**[7조 머신러닝 프로젝트 보고서]**

**"Hand-made MNIST dataset을 이용한 머신러닝 모델 최적화 및 분석"**

**팀장: 전자정보공학부(IT융합전공) 20201610 정하연**

**전자정보공학부(IT융합전공) 20201590 송준규**

**차세대반도체학과 20236079 김희균**

**컴퓨터학부 20192447 정승연**

**[Index]**

**I. 서론**

**1-1. 결과 보고서 요약**

**1-2. 프로젝트 목표**

**II. 본론**

**2-1. 데이터 구하기**

**2-2. 데이터 분석**

**2-3. Dataset 구성 및 테스트 모델 학습**

**2-4. 데이터 전처리 및 파이프라인 구축**

**2-5. 최종 모델 선정 및 모델 학습 진행**

**2-6. 최종 모델 튜닝**

**III. 결론**

**3-1. 피드백**

**3-2. 최종 결론**

**Appendix. 참고 문서**

**I. 서론**

**1-1. 결과 보고서 요약**

~ 모델을 만들었고, ~를 분석했고, ~ 했습니다.

**1-2. 프로젝트 목표**

프로젝트의 첫번째 목표는 Hand-made MNIST dataset과 original MNIST dataset을 이용하여 inference 결과를 비교하고 Hand-made dataset의 성능이 저하된 원인을 밝히고, Hand-made dataset을 포함한 학습데이터를 이용하여 인식 성능을 개선시킨 machine learning model을 학습 시키고 최적화 하는 것입니다.

프로젝트의 두번째 목표는 최종 모델은 Noised Data, Shifted Data 등 학습이나 평가에 방해가 되는 데이터가 존재하더라도 학습과 검증이 잘 되는 내성이 있는 모델을 제작하는 것 입니다.

**II. 본론**

**2-1. 데이터 구하기**

**2-1-1. 프로젝트에 필요한 데이터 정의**

프로젝트에 필요한 dataset을 총 3개로 정의했습니다. 1. original MNIST dataset, 2. hand-made MNIST dataset, original MNIST와 hand-made MNIST를 결합한 3. combined dataset 이렇게 3가지로 정의했습니다. 위와 같은 dataset을 준비하기 위해서는 데이터를 구할 필요가 있었습니다.

**2-1-2. Original / Hand-made 두가지 MNIST 데이터 구하기**

Original MNIST의 경우, Scikit learn의 library를 호출 함으로 구할 수 있었고, hand-made의 경우, 프로젝트를 진행하는 모든 팀의 팀원이 손수 작성하여 제공해 준 데이터를 활용하였습니다. Combined의 경우, 앞선 두가지 데이터만 있으면 조합할 수 있다고 판단하여 다음 단계로 넘어 갔습니다.

**2-2. 데이터 분석**

Original MNIST와 hand-made data를 사용하여 dataset을 구성하기에 앞서서 각각의 데이터는 어떤 특성을 가지고 있는지 확인했습니다. 우선 각각의 data들에 대해서 3가지 사항들을 확인했습니다. 1. 데이터 shape은 어떠한가? 2. 각 데이터가 가진 값은 어떤 값인가? 3. 잘못된 라벨이나 학습이 애매해 보이는 데이터는 없는가? 위의 3가지 기준을 통해 두 데이터를 분석했습니다.

**2-2-1. Original data에 대한 분석**

1. 우선, original MNIST의 경우, image를 784 개의 list로 담고 있다는 것과, 이미지 및 라벨의 수가 70\_000개라는 것을 확인 할 수 있었습니다. 아래 이미지는 Origianl MNIST에 대한 shape를 print한 결과 입니다.



1. 모든 값을 확인 해 볼 수는 없었지만, 특정 N개의 값을 모두 print 해보았을 때, list에 담겨 있는 값이 0.0~255.0 사이의 값을 저장하고 있음을 확인했습니다. 아래 이미지는 Original MNIST의 특정 index에 대한 plot과 저장된 값 중 최솟값과 최댓값을 print한 결과 입니다.

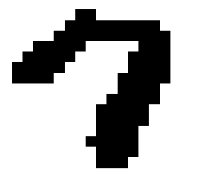
(라벨이 5인 경우)

1. 데이터의 라벨링이 잘못 된 것은 없는가에 대한 분석은 "Official Library인 만큼 라벨링의 오차가 있더라고 그 수는 미비할 것이다" 라는 생각과 70\_000개 라는 숫자를 모두 검증 할 수 있는 시간적 여유가 없었기 때문에 생략했습니다.

**2-2-2. Hand-made data에 대한 분석**(1) Hand-made MNIST의 경우, npz 파일로 부터 load 했을 때 img, label 총 두가지 key를 가지고 있음을 확인했습니다. 각각 해당하는 변수를 선언해 대입한 후, shape를 print 해봤을 때, image를 1차원 list가 아니라 28\*28 로 저장되어 있음을 확인했습니다.



(2)마찬가지로 모든 값을 확인 해 볼 수는 없었지만, 특정 N개의 값을 모두 print 해보았을 때, list에 담겨있는 값이 0.0~1.0 사이의 값을 저장하고 있음을 확인했습니다. 아래 이미지는 Hand-made MNIST의 특정 index에 대한 plot과 저장된 값 중 최솟값과 최댓값을 print한 결과 입니다.

 (라벨이 7인 경우)

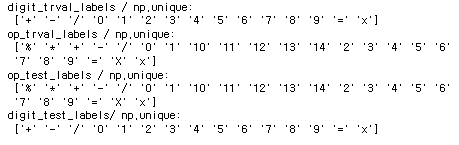
Original MNIST와 다르게 잘못된 라벨이 존재할 가능성이 있어, 모든 데이터를 plot해본 결과 상당 수의 라벨링이 잘못된 데이터와, 식별이 잘 안되는 이미지가 들어 있었습니다. 이에 육안으로 라벨에 해당하는 문자로 보이지 않으면 해당 index를 삭제하는 Data Cleaning 작업이 필요했습니다.

**2-2-3. 두 데이터 분석을 통해 얻은 사실에 의한 통찰**

**(1) 15개(0~9, +, -, x, /, =)클래스 이외의 데이터**

각 데이터의 라벨 차이가 있었습니다. 아래는 Original MNIST와 Hand-made MNIST의 label을 중복이 되지 않게 출력한 결과 입니다. 이를 통해 알 수 있는 사실은 Original과 Hand-made를 결합할 때, Hand-made에 있는 기호가 Original에는 것과, 원래 의도 했던 0~9, +, -, x, /, =를 제외 하고 더 많은 잘못된 라벨이 들어 있다는 사실입니다.

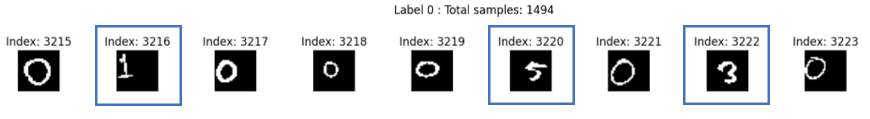




해당 사실들을 바탕으로 추후 Prepare Dataset 단계에서 의도 되지 않은 라벨은 제거할 계획을 수립하였습니다.

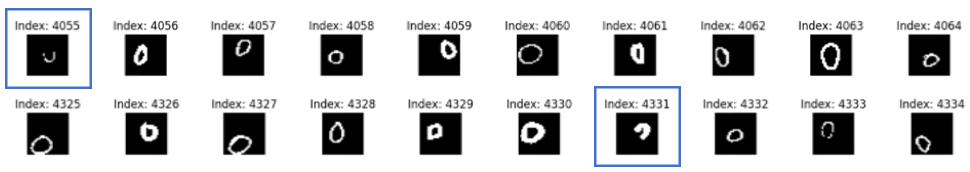
**(2) 잘못된 라벨링**

클래스 별 데이터의 특징을 알아보기 위해 15개의 클래스 순서대로 데이터를 나열했습니다. 0으로 라벨링된 데이터, 1로 라벨링된 데이터, 2로 라벨링된 데이터, … 와 같은 순서대로 나열한 것입니다.



위는 0으로 라벨링된 데이터 10개를 임의로 plot한 결과입니다. 3216, 3220, 3222 데이터는 육안으로 보면 각각 1, 5, 3에 해당하는 데이터인데, 0으로 잘못 라벨링 되었습니다.

**(3) 학습에 애매한 데이터**



위는 0으로 라벨링된 또 다른 10개의 데이터를 plot한 결과입니다. 4055 데이터는 0의 위쪽 부분이 잘려 숫자 0의 형태라고 보기 어렵습니다. 4331 데이터는, 4330 데이터와 같은 굵기와 크기를 가져서 0의 왼쪽 아래부분이 잘렸다고 볼 수 있지만, 클래스 7로도 분류될 수 있는 이미지입니다. 이와 같이 라벨링은 제대로 되었지만 심하게 잘린 데이터, 혹은 다른 클래스와 유사하여 혼동을 유발할 수 있는 데이터가 존재함을 알게 되었습니다.

**(4) Handmade dataset 데이터 수**

약 60\_000개의 Original MNIST dataset과 달리, Handmade MNIST dataset은 약 30\_000개만 존재함을 알게되었습니다. 추후 handmade dataset 전처리 과정을 거친 후, original MNIST dataset를 일부 추가하여 데이터 개수에 보완이 필요함을 알게 되었습니다.

**(5) 픽셀값(feature 값)의 차이**

Original MNIST dataset의 픽셀값은 0에서 255 사이의 값이고, Handmade MNIST dataset의 픽셀 값은 0과 1 사이의 값이었습니다. Handmade 와 Original MNIST dataset을 통합한 dataset을 학습하려면, 픽셀 값을 통일하는 (스케일링) 과정이 필요함을 알게 되었습니다.

**(6) Noise가 있는 데이터, Shift된 데이터, 테두리가 남은 데이터**



현재 가지고 있는 테스트 데이터셋 이외의 어떠한 데이터가 들어와도 강력한 모델을 만들기 위해서는, 노이즈가 있는 학습 데이터셋을 구축하는 작업이 필요함을 인지하게 되었습니다.

이를 위해서는 노이즈로 분류되는 threshold를 정의해야 하고, 모델이나 함수 중 어떤 방식으로 노이즈를 처리할 지 결정하는 과정이 필요했습니다. 또한, Index 519와 같이 handmade dataset을 작성할 때 썼던 템플릿의 테두리가 포함된 데이터가 존재함을 알게 되었고, 학습에 방해가 될 수 있다고 판단했습니다.

index 520, 522와 같이 숫자가 중앙에서 벗어나 상하좌우로 치우친 데이터도 존재했습니다. 데이터의 일관성이 떨어지므로 중앙화를 하거나 상하좌우로 치우친 데이터를 추가로 제작하는 과정이 필요하겠다는 생각을 하게 되었습니다. 그리고 최종적으로 test dataset에도 관련 데이터를 구축하여 검증할 필요가 있다고 판단하였습니다.

**(7) 비슷한 숫자 및 기호가 존재**

****

Index 3888과 같이 0이지만 자칫 6으로 판별될 수도 있는 애매한 데이터가 존재하기에, confusion matrix 분석을 통해 가장 많이 혼동되는 라벨을 파악할 필요가 있다고 생각했습니다. Index 3866과 같이 기울어진 이미지도 변수로 두어 기울어진 정도를 달리한 데이터셋을 구축하는 아이디어도 생각해 보았지만, 기울기라는 변수는 1과 /를 식별하는 중요한 feature를 방해할 수 있다고 판단해 이후 전처리에서는 다루지 않기로 결정했습니다.

**(8) 데이터의 크기**

현재 Original, Handmade MNIST dataset 모두 28\*28 크기인데, 다른 크기를 입력으로 받아도 처리가능한 모델을 만들 수 있겠다는 아이디어 또한 도출되었습니다.

**2-3. Dataset 구성 및 테스트 모델 학습**

앞서 프로젝트에 필요한 데이터 셋을 1. original MNIST dataset, 2. hand-made MNIST dataset, original MNIST와 hand-made MNIST를 결합한 3. combined dataset 이렇게 3가지로 정의했습니다. 이를 위해 모은 Data를 결합하여 총 4개의 Dataset을 구성했습니다.

Original MNIST dataset은 학습 데이터를 80%, 검증 데이터를 20%로 나누었습니다.

Handmade MNIST dataset은 아래 그림과 같이 구축했습니다. 우선 약 30000개의 제공된 데이터 중 80%는 학습 데이터, 20%는 검증 데이터로 사용하였습니다. 그리고 향후 파이프라인에 추가한 중앙화 및 최대화 작업이 제대로 수행되는지 수치적으로 명확하게 확인하기 위해서, 모든 데이터를 중앙화 시킨 후 전체 데이터 중 20%는 shift 시켰습니다. Left Top, Left Bottom, Right Top, Right Bottom으로 숫자나 기호를 이동시켰습니다. 그리고 training과 validation dataset 각각의 25%에 해당하는 데이터에 노이즈를 추가했습니다.

A diagram of a graph

Description automatically generated with medium confidence

노이즈를 추가는 다음과 같이 진행했습니다. 예를 들어 픽셀이 0에서 1 사이의 값을 가진다고 하면, 0과 0.5 사이의 랜덤 값을 784개 픽셀이 각각 더했습니다. 그래서 향후 파이프라인의 denoising 과정에서 픽셀 값이 약 0.5~1.0 이상의 값일 때 유의미한 픽셀로 인식했습니다.

Combined Dataset 또한 데이터 수가 추가되었을 뿐, 비율은 동일하게 설정했습니다. 각 숫자 클래스 당 데이터 개수가 같도록 Original Dataset을 추가했습니다. 기호 데이터는 Original Dataset으로 보완할 수 없기에, 각각 같은 데이터 수를 가지도록 이미 가지고 있는 데이터를 복제하여 개수를 늘렸습니다. 최종적으로 학습데이터 59612개와, Validation Dataset 9900개로 구성되었습니다.

Fine-Tuning 후 테스트에 사용할 Final Test Dataset 또한 동일한 비율로 shift하고, 노이즈를 생성했습니다.

**2-4. 데이터 전처리 및 파이프라인 구축**

앞서 서술한 데이터에 대한 통찰의 결과로, 파이프라인을 크게 5단계로 구축했습니다.

첫 번째는 Denoise 과정입니다. 모델의 테스트 결과에서 노이즈가 심한 데이터가 들어올 수 있으며, 저희는 이에 강건한 모델을 만들기 위하여 노이즈 제거 함수를 만들어 파이프라인 첫 번째에 넣었습니다. 임계값을 0.4로 설정하여 픽셀 값이 0.4~1.0 이상의 값만 의미있는 값으로 판별되도록 denoise\_with\_max 함수를 정의했습니다. 초기에는 이론적으로 가장 strict 한 0.66으로 설정했지만, 다양한 임계값을 시도해본 결과 0.4 일 때 가장 노이즈를 효과적으로 제거함을 확인하였습니다.

두 번째는 Image Centering (중앙화) 과정입니다. 처음에는 shift된 데이터를 다루기 위해 상하좌우로 치우쳐진 데이터를 일정 비율 생성하고 이 데이터를 넣어 학습시키려는 계속이었습니다. 교수님께서 상하좌우로 치우친 데이터는 큰 의미가 없다는 조언을 해 주셨고, 360 도로 100 개 정도 다른 위치에 존재하는 데이터를 구축하지 않는 이상 파이프라인에 중앙화 과정을 넣는 것이 효과적임을 알게 되었습니다. 그래서 아래 어떤 데이터가 들어오든 숫자나 기호를 중앙에 위치시킬 수 있는 과정을 구축하였습니다.

세 번째는 Image Enlarging (최대화) 과정입니다. 기호를 중앙에 두어도 데이터 작성자에 따라 숫자나 기호의 크기가 매우 상이했습니다. 그래서 저희는 조금 더 명확하게 숫자나 기호를 인식하기 위해 이미지를 최대화하는 과정을 추가했습니다. 최대화 과정을 거치면 아래와 같은 이미지로 변환이 됩니다.

A black and white image of a logo

Description automatically generated

네 번째는 Normalize (정규화) 과정입니다. 앞서 언급했듯 Original Dataset의 픽셀 값은 0~255 사이이고, Handmade Dataset의 픽셀 값은 0 또는 1입니다. 두 데이터를 합쳐 Combined Dataset을 만드는 과정에서 어떤 픽셀 범위가 섞여서 들어오든 이에 강건한 모델을 만들어야겠다는 생각이 들어 정규화 과정을 추가하게 되었습니다. 어떤 정규화 방식을 사용할 지 결정하기 위해 간단한 KNN 모델에 다양한 정규화 방식을 적용하여 Accuracy를 측정해보았습니다.

A screenshot of a computer

Description automatically generated

테스트 결과 Normalizer()를 사용할 때 가장 정확도가 높았기에 해당 방식을 사용했습니다. 그리고 Normalizer의 norm를 l2로 설정할 지, max로 설정할지 고민하는 과정이 있었습니다. Max로 설정했을 때 픽셀 값의 최대가 1로 설정되기에 노이즈를 추가하는 식을 작성하는데 용이할 것이라 생각했지만, ~~~~~~~~~~~~~

마지막 단계는 모델입니다. ~~~~~~~~~~~~~

**2-5. 최종 모델 선정 및 모델 학습 진행**

**2-5-1. 프로젝트에 필요한 모델 정의**

Original Dataset로 KNN, SVM, Extra-Tree, Softmax, Decision Tree, Random Forest, MLP를 학습시킨 후, 모든 모델을 VotingClassifier에 넣어 Accuracy를 확인했습니다. SVM보다 오히려 낮은 Accuracy였고, VotingClassifier는 각각의 모델 학습 시간까지 합하면 학습에 너무 많은 시간이 소요되기에 효율적이지 않은 모델이라고 판단하여 사용하지 않기로 결정했습니다.

A screenshot of a graph

Description automatically generated

Handmade Dataset으로 나머지 모델들을 GridSearch로 학습하고 검증한 결과, Softmax와 Decision Tree의 성능이 낮아서 사용하지 않기로 결정했습니다.

A screenshot of a graph

Description automatically generated

**2-6. 최종 모델 튜닝**

**III. 결론**

**3-1. 피드백**

* 정규화 모델 설정

**3-2. 최종 결론**

로 결론을

**Appendix. 참고 문서**

" Local Area Networks (VLAN\_802.1Q)"